

DLPROTEIN 1.2 User Manual

Contents

Overview	1
1 Building the molecular topology	3
1.1 <u>dlgen</u>	5
1.1.1 <u>dlgen</u> input files	6
1.1.2 Editing the pdb file	6
1.1.3 Adding entries to the MBBT file	10
1.1.4 Adding entries to the PS file	11
1.2 <u>merge_field</u>	14
2 The molecular dynamics code	16
2.1 Integration Algorithms	16
2.1.1 NVT ensemble	17
2.1.2 NPT ensemble	19
2.1.3 Constraints treatment	20
2.2 RESPA implementation	21
2.3 Scanning neighbors	24

2.3.1	Memory optimization on scalar workstation	24
2.3.2	CPU optimization on scalar workstation	25
2.4	The Smooth Particle Mesh Ewald method	26
2.4.1	Choosing the SPME variables	27
3	Other differences between DLPOLY and DLPROTEIN	29
3.1	Non-bonded interactions	29
3.2	Interpolation, spline or direct evaluation of the Van der Waals in- teractions	29
3.3	Refolding of molecules	30
3.4	Constraints treatment	30
3.5	Shifted Van der Waals interactions and long range corrections . . .	31
3.6	Smoothing the Van der Waals interactions at the cut-off	31
3.7	Hydrogen bond detection	32
3.8	Short and long OUTPUT files	32
3.9	I/O format of trajectory data	32
3.10	Include files	33
3.11	T3E communication routine	33
3.12	Directives in the CONTROL file	34
3.13	Directives in the FIELD file	36
3.14	Directives in the Makefile	37
3.15	Examples	39
	References	41

Overview

DLPROTEIN is a molecular dynamics package written by Simone Melchionna and Stefano Cozzini in the framework of the Italian Institute for the Physics of Matter (**INFN**), Network on “MD simulation of biosystems”. DLPROTEIN is a development of the original general purpose Molecular Dynamics code **DLPOLY** written at Daresbury Lab (UK) by William Smith and Tim R.Forester.

The motivation underlying the development of DLPROTEIN has been to specialize the original MD code to deal more specifically with biological molecules, with particular focus on proteins, and with high efficiency and low memory requirements for scalar architectures.

DLPROTEIN has been realized with the contributions of several people who have participated in the project in different percentages. Amongst others we wish to thank Antonella Luise as a coauthor of dlgen and for designing the link cell algorithm for generic periodic boundary conditions, Maddalena Venturoli as a coauthor of dlgen and for many useful insights in the code, Marco Pierro, as the author of the merge_field utility, from the Physics Department of the University “La Sapienza” of Rome. A special thanks is for Georges Destree, of the Université Libre de Bruxelles, who has managed several parallel fixes and new algorithms in the code.

During the development of DLPROTEIN collaboration and support from Giovanni Ciccotti (University of Rome “La Sapienza”), Mattia Falconi, Alessandro Desideri (University of Rome “Tor Vergata”) and Mauro Ferrario (University of

Modena) are kindly acknowledged. We thank Tom Darden for providing us with the SPME original routines.

The following manual is intended as an addendum to the DLPOLY 2.0 Manual [1]. The reader is referred to the DLPOLY manual for general explanations about the code philosophy and technical details whereas the present documentation reports only modifications and addenda to the original DLPOLY package.

Chapter 1

Building the molecular topology

Building the molecular topology of molecules is done by the utility build_all . The program build_all takes care of building up in sequence a system composed of one or more molecules. build_all is a fortran program that calls the two different utilities called dlgen and merge_field .

The three programs build_all , dlgen , and merge_field are provided with a Makefile in the **build-topology** directory. Once they are compiled (by simply executing the “**make**” command), these utilities can be run altogether, by running the driver build_all , or singularly.

dlgen converts a Protein Data Bank file (pdb file) into the files CONFIG and FIELD that contain respectively the topology and configuration of a single molecule. Subsequently merge_field merges each FIELD file produced by dlgen , eventually with other prebuilt FIELD files, into a global FIELD file. These utilities have been implemented thinking in terms of aminoacidic systems (e.g. proteins) as the main goal, and it is able to treat force fields of the type **CHARMm22** and **GROMOS87** (**37c4** parameter set). Due to the general combination rules of the CHARMm22 force field, dlgen works for proteins, proteins+metallic groups (e.g. heme group), phospholipids, sugars, and so on, i.e. any kind of entry can be safely added to the molecular database. Viceversa due to the more complex rules

adopted by the GROMOS87 force field (e.g. the way the dihedrals are defined for sugars), the actual version of dlgen works only for proteins, eventually in presence of metallic groups. The reader may ask the authors to obtain a non-standard version of dlgen to treat different systems within the GROMOS87 force field.

An easy way to run the build_all utility is to make the following steps :

1. Link (or Copy) in your working directory the three executables produced by the compilation, together with the database files (MBBT and PS files, as explained in the following) needed by dlgen to run. In the working directory you need to have also the PDB files to be converted or any previously produced FIELD file the needs to be merged.
2. Run the build_all utility and answer a few questions. Infos and warnings will be issued as the molecular construction proceeds.

The dlgen utility supports the “United Atom” (**UA**) scheme (i.e. CH, CH₂ and CH₃ group are treated as united groups) when using the GROMOS87 force field, and the “All Hydrogens” (**AH**) scheme (polar and non-polar hydrogens are always explicitly treated) when using the CHARMM22 force field.

dlgen uses an internal database to store the parameter set (**PS**) and the molecular building block topology (**MBBT**), e.g. the structure of the elementary units, such as the amino acids. The PS file stores:

- atomic names, masses and connection tables;
- bond reference list, spring constants and equilibrium distances;
- angle reference list, spring constants and equilibrium angles;
- dihedral reference list, multiplicity, force constants and phases;
- Van der Waals parameters.

The MBBT files stores the group names, internal connectivity and atomic charges. In table 1.1 we report the database file names.

	PS file name	MBBT file name
GROMOS87	PS_gromos37c.ff	MBBT_gromos37c.ff
CHARMM22	PS_charmm22.ff	MBBT_charmm22.ff

Table 1.1: Database file names for the GROMOS87 and CHARMM22 force fields. PS : parameter set, MBBT : molecular bulding block topology

1.1 dlgen

The dlgen utility must be run with a pdb file that specifies a single molecule, i.e. each atom of the molecule must be globally connected by chemical bonds.

dlgen performs in order the following actions :

1. Read in the Brookhaven pdb file and the MBBT files.
2. Generate the sequence of each block by merging the informations in the pdb and MBBT files (**edtgen** routine).
3. If hydrogen atoms are missing, add them with the proper chemical geometry and connectivity, as specified in the MBBT database (**hgen** routine).
4. Read in the PS and FF.dat files.
5. Complete connectivity by adding the missing links (e.g. link aminoacid to aminoacid, complete the aromatic rings, and so on).
6. Generate bond, angle and dihedral list, together with the Van der Waals interactions (generic and I-IV types).
7. Dumps out the FIELD file.

1.1.1 dlgen input files

dlgen uses the following input data :

- **Brookhaven pdb** file, containing atomic and molecular names and atomic configuration. This file must be edited and modified, as explained later on.

- **FF.dat** file, containing the topology title, constraints-vs.-harmonic springs treatment and energy units. The constraints directives are :

directive:	meaning:
constrain <i>string</i>	where <i>string</i> can be equal to <i>all</i> : all bonds are constrained <i>h</i> : only bonds concerning hydrogens are constrained <i>none</i> : all bonds are harmonic
specialbond n_1 n_2 n_3 ...	list of residues to treat harmonically.
units <i>string</i>	Output units as expressed in the FIELD file. Admitted values for <i>string</i> are “eV”, “kJ”, “kcal” and “internal’ for using dlpoly internal units.

Moreover the user must insert data as requested by the program in run time at standard input.

1.1.2 Editing the pdb file

Before running the dlgen program the pdb file must be edited and a few changes must be applied. dlgen uses each building block specified in the MBBT database as an independent unit so that in principle there is no difference when treating a sequence of amino acids or a sequence of generic building blocks. In fact the intrablock topology is specified in the MBBT database, while dlgen uses the geometry specified in the pdb file to construct the interblock topology.

1. Terminal groups labels

Usually the peptidic sequence begins (or ends) with a NH_3 group. The pdb atomic label relative to the starting nitrogen atom must be changed from the original pdb label into the new NTER label and eventually if

any hydrogen is specified (i.e. the user does not require generation of positions hydrogens from scratch) the hydrogen atoms must have the labels **HTn**, with **n**=1, 2, 3. Hydrogen atoms attached to **NTER** must follow in order. In case the N-terminal aminoacid is a Proline, the name of the starting nitrogen should be left as **N** and the aminoacid name should be changed from **PRO** to **PRT**.

When the peptidic sequence has *COO* as a terminal group, the label of the carbon atom must be changed into the **CTER** new label and the oxygens must have the new labels **OTn**, with **n**=1, 2. Oxygens attached to **CTER** must follow in order.

When having terminal **PRO** residues, the user must define from scratch the complete terminal residue (**PRO**+terminal group) in the MBBT file as it is specified in the section about adding entries to the MBBT file.

2. Non aminoacidic specification

In the standard pdb format, non aminoacidic atoms have as special molecular label **HETATM** instead of **ATOM**. The user must respect this labeling when specifying groups as prosthetic groups, such as heme molecule or single metals that are specified as separate entities in the MBBT file.

3. Atom and block indexing

The pdb atomic indexing is ignored by dlgen . Viceversa the building block (e.g. residue) indexing is used to signal the change in the molecular building block, although the order of appearance is unimportant.

4. Protonation state and disulfide bridges

Residues that in the pdb file have the same names but intrinsic differences, e.g. different protonation states due to different pK's or disulfide bridges, must be named consistently with the database names in the MBBT file so that they adhere with the corresponding database definitions.

In the following example, it is indicated how to change the head and tail specification of a peptide. Given the original pdb file:

```
ATOM      1  N   ALA      1      15.850  25.550  23.340
ATOM      2  HT1 ALA      1      15.520  25.140  24.200
ATOM      3  HT2 ALA      1      15.700  26.540  23.360
ATOM      4  HT3 ALA      1      15.360  25.150  22.570
ATOM      5  CA  ALA      1      17.270  25.280  23.200
...
ATOM    1188  HZ3 LYH     128      31.830  21.930  16.050
ATOM    1189  C   LYH     128      27.000  27.260  13.390
ATOM    1190  OT1 LYH     128      26.420  28.360  13.350
ATOM    1191  OT2 LYH     128      26.460  26.250  12.940
...
```

You need to change it into :

```
ATOM      1  NTERALA%%      1      15.850  25.550  23.340
ATOM      2  HT1 ALA      1      15.520  25.140  24.200
ATOM      3  HT2 ALA      1      15.700  26.540  23.360
ATOM      4  HT3 ALA      1      15.360  25.150  22.570
ATOM      5  CA  ALA      1      17.270  25.280  23.200
```

```

. . .
ATOM    1188  HZ3  LYH    128      31.830  21.930  16.050
ATOM    1189  CTERLYH    128      27.000  27.260  13.390
ATOM    1190  OT1  LYH    128      26.420  28.360  13.350
ATOM    1191  OT2  LYH    128      26.460  26.250  12.940
. . .

```

1.1.3 Adding entries to the MBBT file

In the MBBT database the first lines concern the gromos vs charmm signal, title and number of subsequent block entries.

For each entry the format is specified as in the following example for the alanine residue :

```
residue = ala
atoms = 6
  1    N    N   -0.28
  2   1H    H    .28
  3   1CA  CH1   .0
  4   3CB  CH3   .0
  5   3C    C    .38
  6   5O    O    - .38
```

The first column (i4) is the atom indexing, second column (i4) is the index of the preceeding atom linked to the current atom. The general rule is that each attachment specifies only links with previously defined atoms. If the second column index is omitted, since it concerns interblock connectivity, or if the complete connectivity cannot be specified, as is the case for aromatic rings, dlgen builds the connectivity by using geometric rules, as coordinates are extracted from the pdb file.

The third column (a4) regards the pdb atomic names, while the fourth column (a4) specifies the atom type name (coherently with the atomic names as specified in the PS file). The fifth column (f8.0) specifies the atomic charge. The last column (f8.0), if present, specifies the repetition of the current atom data.

The “nte” and “cte” block names as fictitious names and they need not to

be considered as new residue names, as they are needed only to specify data of the terminal groups.

Some specific patches can be added to the MBBT file. These patches can be necessary if dlgen fails to build the full topology of complex molecules. The patches are used to add or delete specific constraints, angles bendings (harmonic type), dihedral torsions (cosine type), improper torsions (harmonic type) and bonds (harmonic types) potential terms from the final topology. The general format of each patch should be in the form :

```
add keyword num
```

```
list(1,...,n) par(1,m) (in free format)
```

or

```
del keyword num
```

```
list(1,...,n) par(1,m) (in free format)
```

where **keyword** can be **constraint**, **angle**, **dihedral**, **improper**, **bond**. **num** is the number of patches specified subsequently. **list(1,...,n)** is the list of atomic indices involved in the patch indexed as the atoms appearance in the block itself. **par(1,...,m)** is the list of parameters that need to be fixed (appearing in the same order of the FIELD file) in the same units as specified previously. An example of patch can be found in the MBBT block specifying the HEME group (keyword "hem").

1.1.4 Adding entries to the PS file

The first line of this file indicates the force field type (charmm or gromos). Moreover, a line is skipped if starts with the character "**#**" while each new entry is indicated by the special string **BLOCK**. The order of appearance of the **BLOCKs** must be the same as the PS file included in the distribution. Atomic labels are consistent with those specified in the fourth column of the MBBT file.

The database contains the following data:

- Number of records in the atom list
- Atom names (a4), masses (f12.0) and repetitions (i5)
- Number of records in the bonds list
- Prefactor for scaling the Van der Waals data between pairs
- Bond pair names (2a4), force constants (f12.0) and equilibrium distances (f12.0)
- Number of records in the angle bends list
- Prefactor for scaling the Van der Waals data between atomic triplets
- Angle triplet names (3a4), force constants (f12.0) and equilibrium angles (f12.0)
- Number of records in the general proper dihedrals (GPD) list
- Prefactor for scaling the Van der Waals data between atomic quadruplets of the GPD list
- Potential type (cos), scaling factor for electrostatics and Van der Waals interactions
- GPD quadruplet names (4a4), force constants (f12.0), phase (f12.0) and multiplicities (f12.0).
- Number of records in the specific proper dihedrals (SPD) list
- Prefactor for scaling the Van der Waals data between atomic quadruplets of the SPD list

- Potential type (cos), scaling factor for electrostatics and Van der Waals interactions
- SPD quadruplet names (4a4), force constants (f12.0), phases (f12.0) and multiplicities (f12.0).
- Number of records in the general improper dihedrals (GID) list
- Prefactor for scaling the Van der Waals data between atomic quadruplets of the GID list
- Potential type (harm), scaling factor for electrostatics and Van der Waals interactions
- GID quadruplet names (4a4), force constants (f12.0), and equilibrium angles (f12.0).
- Number of records in the specific proper dihedrals (SID) list
- Prefactor for scaling the Van der Waals data between atomic quadruplets of the SID list
- Potential type (harm), scaling factor for electrostatics and Van der Waals interactions
- SID quadruplet names (4a4), force constants (f12.0), and equilibrium angles (f12.0).
- Number of records in the general Van der Waals parameters list
- General Van der Waals parameters
- Number of records in the specific Van der Waals parameters list
- Specific Van der Waals parameters

- Number of records in the general 1..4 Van der Waals parameters list
- General 1..4 Van der Waals parameters
- Number of records in the specific 1..4 Van der Waals parameters list
- Specific 1..4 Van der Waals parameters

Here “general dihedrals” (both proper or improper) specifies the quadruplets having specified a reduced number of atoms defining the dihedral. The wild card symbol “*” indicates all possible atom types included in the quadruplet. On the other hand “specific dihedrals” means the quadruplets explicitly indicated.

Similarly “general” and “specific” Van der Waals interactions (for 1..4 interactions or not) refer to the parameter set used for all kind of atomic pairs and for selected pairs respectively. The 1..4 parameters specify the interaction parameters between pairs of atoms that are third neighbours in the molecular connectivity.

1.2 merge_field

This utility allows to merge different FIELD files into a single FIELD file, called FIELD.out, where each starting FIELD file must contain one molecular specie. Moreover each FIELD file must specify the Van der Waals parameters in “12-6” style.

When running merge_field , the user is requested to provide as standard input the force field scheme (charmm or gromos), the name of each FIELD file, together with the relative MBBT file and the number of repetitions of each molecular specie. If the name of some atom in one FIELD file coincides with the name in a second FIELD files and those atoms have different interaction parameters, the user is requested to change the name of atoms in the second FIELD file. This can be done by the utility itself or by editing one of the FIELD files. The utility can

merge as many molecular species as needed by changing the `mxmoltyps` parameter in the `merge_field.inc` file and recompiling. The output files “VDW_params” and “SYSTEM” can be used by the user to check if the merging operation was correctly achieved.

Chapter 2

The molecular dynamics code

2.1 Integration Algorithms

In DLPROTEIN the algorithm that integrates in time the equations of motion is of the form of the Velocity Verlet (VV) scheme whereas in DLPOLY the original scheme was the Leap Frog (LF) scheme. In addition the VV scheme is applied in the form proposed in ref. [4] so to have, even in presence of non-hamiltonian dynamics (as it is customary for generating canonical or isobaric ensembles), time reversible and symplectic integrators.

Given the Hamiltonian equations of motion

$$\begin{aligned}\dot{r} &= \frac{p}{m} \\ \dot{p} &= F\end{aligned}\tag{2.1}$$

where velocities are related to momenta via the relation $p = mv$, the LF scheme integrates the trajectory in time with the following updating scheme

$$\begin{aligned}r(t+h) &= r(t) + hv(t+h/2) \\ v(t+h/2) &= v(t-h/2) + \frac{h}{m}F(t)\end{aligned}\tag{2.2}$$

together with the auxiliary equation to obtain velocities simultaneously to posi-

tions

$$v(t) = \frac{1}{2} [v(t - h/2) + v(t + h/2)] \quad (2.3)$$

The VV scheme, that for hamiltonian system produces exactly the same trajectory than the LF one, reads

$$\begin{aligned} r(t+h) &= r(t) + hv(t) + \frac{h^2}{2m} F(t) \\ v(t+h) &= v(t) + \frac{h}{2m} [F(t) + F(t+h)] \end{aligned} \quad (2.4)$$

The VV scheme can be casted in operatorial form by using the Trotter factorization of the evolution operator [4]

$$\begin{aligned} \exp(ihL_{Ham}) &= \exp\left(i\frac{h}{2}L_v\right) \exp(ihL_r) \exp\left(i\frac{h}{2}L_v\right) \\ &= \exp\left(i\frac{hF}{2m}\frac{\partial}{\partial v}\right) \exp\left(ihv\frac{\partial}{\partial r}\right) \exp\left(i\frac{hF}{2m}\frac{\partial}{\partial v}\right) \end{aligned} \quad (2.5)$$

Each operator is applied sequentially and gives rise to a shift

$$\exp\left(a\frac{\partial}{\partial x}\right)f(x) = f(x+a) \quad (2.6)$$

In DLPROTEIN the single-time-step molecular dynamics is handled by the routine **sloop**, and the action of the propagators is performed by the routines

$$\begin{aligned} \text{respa_v_verlet} &\leftrightarrow \exp\left(i\frac{h}{2}L_v\right) \\ \text{respa_r_verlet} &\leftrightarrow \exp(ihL_r) \end{aligned}$$

applied according to the Trotter splitting. The same routines are called when performing NVT and NPT simulations with slight modifications for the NPT case as explained in the following.

2.1.1 NVT ensemble

DLPROTEIN implements the canonical ensemble and the isobaric-isothermal ensemble with the Nosé-Hoover style equations of motion[5]. Other kinds of statistical behaviours, such as the the Evans and Berendsen schemes can still be used.

The Nos-Hoover NVT equations of motion read

$$\begin{aligned}\dot{r} &= \frac{p}{m} \\ \dot{p} &= F - \chi p \\ \dot{\chi} &= \frac{1}{\tau_T^2} \left(\frac{T}{T_{ext}} - 1 \right)\end{aligned}\tag{2.7}$$

Accordingly the integration scheme is given by the splitting

$$\exp(ihL) = \exp\left(i\frac{h}{2}L_{NH}\right) \exp(ihL_{Ham}) \exp\left(i\frac{h}{2}L_{NH}\right)\tag{2.8}$$

where

$$iL_{NH} = -\chi p \frac{\partial}{\partial p} + \frac{1}{\tau_T^2} \left(\frac{T}{T_{ext}} - 1 \right) \frac{\partial}{\partial \chi}\tag{2.9}$$

For the NVT and NPT dynamics we make use of the general formula when applying the effect of the thermostat and piston on the momenta and positions

$$\exp\left[(ax+b)\frac{\partial}{\partial x}\right] = \exp(a)x + b \exp(a/2) \frac{\sinh(a/2)}{(a/2)}\tag{2.10}$$

The $\sinh(x)/x$ function is computed by expanding numerator and denominator to a 8th order Taylor expansion.

The Nosé-Hoover propagator is further symmetrically splitted in order to deal with intrinsic momenta dependence of the thermostat driving force. The splitting is of the type

$$\begin{aligned}\exp(i\frac{h}{2}L_{NH}) &= \exp\left[\frac{h}{4}\frac{1}{\tau_T^2} \left(\frac{T}{T_{ext}} - 1 \right) \frac{\partial}{\partial \chi}\right] \\ &\quad \exp\left[-\frac{h}{2}\chi p \frac{\partial}{\partial p}\right] \exp\left[\frac{h}{4}\frac{1}{\tau_T^2} \left(\frac{T}{T_{ext}} - 1 \right) \frac{\partial}{\partial \chi}\right]\end{aligned}\tag{2.11}$$

The effect of the Nosé-Hoover propagator is given by the routine

$$\mathbf{prop_NH} \leftrightarrow \exp\left(i\frac{h}{2}L_{NH}\right)\tag{2.12}$$

The same routine is applied for the NPT case with minor modifications.

2.1.2 NPT ensemble

The NPT Nosé-Hoover equations of motion are

$$\begin{aligned}
\dot{r} &= \frac{p}{m} + \eta(r - R_o) \\
\dot{p} &= F - (\chi + \eta)p \\
\dot{\chi} &= \frac{1}{\tau_T^2} \left(\frac{T}{T_{ext}} - 1 \right) \\
\dot{\eta} &= \frac{1}{3Nk_B T_{ext} \tau_P^2} V (P - P_{ext}) \\
\dot{V} &= 3\eta V
\end{aligned} \tag{2.13}$$

where R_o is the total center of mass of the system. The Trotter splitting is chosen as

$$\exp(ihL) = \exp\left(i\frac{h}{2}L'_{NH}\right) \exp(ihL'_{Ham}) \exp\left(i\frac{h}{2}L'_{NH}\right) \tag{2.14}$$

where

$$\begin{aligned}
iL'_{Ham} &= iL_{Ham} + \eta(r - R_o) \frac{\partial}{\partial r} + 3V\eta \frac{\partial}{\partial V} \\
iL'_{NH} &= -(\chi + \eta)p \frac{\partial}{\partial p} \\
&+ \frac{1}{\tau_T^2} \left(\frac{T}{T_{ext}} - 1 \right) \frac{\partial}{\partial \chi} + \frac{1}{3Nk_B T_{ext} \tau_P^2} V (P - P_{ext}) \frac{\partial}{\partial \eta}
\end{aligned} \tag{2.15}$$

The NPT Nosé-Hoover propagator is further symmetrically splitted in order to deal with intrinsic momenta and coordinates dependence of the thermostat and piston driving forces. The splitting is analogous to equation 2.11. The effect of the NPT Nosé-Hoover propagator is again given by the routine

$$\mathbf{prop_NH} \leftrightarrow \exp\left(i\frac{h}{2}L'_{NH}\right) \tag{2.16}$$

The NST ensemble equations of motion, i.e. the dynamics associated with anisotropic volume fluctuations, are not yet implemented in the current release.

For the Berendsen thermostat, the evolution propagator has been splitted in an analogous way to the Nosé-Hoover way. The routine that scales the velocities according to the Berendsen scaling factor is **prop_ber**.

The Evans constraint on velocities, that allows to sample the isokinetic ensemble, is achieved in the routine **evans_scale**.

2.1.3 Constraints treatment

In the VV scheme treatment of the holonomic constraints is handled in a similar way to the LF scheme. The main difference relies in imposing simultaneously the condition of satisfied constraints

$$\sigma_k(\{r_i\}) = 0 \quad k = 1, M \quad (2.17)$$

as handled by the routines **rdshake_1** and **sca_shake** (SHAKE procedure) and condition that momenta are orthogonal to the constraints surfaces

$$\dot{\sigma}_k(\{r_i, p_i\}) = 0 \quad (2.18)$$

as handled by the routines **quench** and **quench1** (known as the RATTLE procedure). We have modified the preexisting **rdshake_1** routine into the new **sca_shake** routine. The change has been made in order to solve the iterative sequence of constraints by cyclic update of atomic positions [3]. At the moment this procedure is possible only on scalar architectures. Viceversa the previous **rdshake_1** is applied on the full sequence of constraints in a different way and it has been motivated by parallelisation reason [1]. The **sca_shake**, although running on scalar platforms, converges in about half the iterations than the parallel version. In an analogous way the new **sca_quench** routines uses a cyclic update of atomic velocities.

For the NVE and NVT ensembles the sequence of propagators and constraints impositions is

$$\exp\left(i\frac{h}{2}L_{NH}\right) \mathbf{R} \exp\left(i\frac{h}{2}L_v\right) \mathbf{S} \exp(ihL_r) \exp\left(i\frac{h}{2}L_v\right) \exp\left(i\frac{h}{2}L_{NH}\right) \quad (2.19)$$

where **R** and **S** are the operators that handle constraints on velocities and positions respectively.

In the NPT implementation the overall procedure is modified in order to deal with intrinsic dependence of the pressure tensor on the constraining forces, as discussed in [4]. In this case an iteration loop need to be applied in order to converge the pressure estimate and piston update so to achieve proper integration in time. Convergence of the piston updating is controlled by the routine **checkexp**.

For the Berendsen isothermal + isobaric dynamics, the splitting of the propagator is rather crude, but it seems to produce a statistical behaviour rather similar to the Berendsen implementation of the original DLPOLY package.

2.2 RESPA implementation

A multiple time step implementation, generally known as the RESPA method, has been applied in the framework of the VV scheme [4] and within a splitting of the Hamiltonian well suited for biological systems [6]. The propagator is in the form

$$\exp(ihL) = \exp[ih(L_o + L_1 + L_2 + L_3)] \quad (2.20)$$

where L_o takes into account positions shifts and updating of momenta according to bond stretching and angle bending forces. L_1 shifts momenta due to dihedral torsions and first-shell non-bonding interactions. L_2 shifts momenta due to second-shell non-bonding interactions and reciprocal Ewald term. L_3 shifts momenta due to third-shell non-bonding interactions [6]. First, second and third non-bonding forces are defined by introducing three interaction ranges

$$\begin{aligned} 0 &< r_{ij} < \mathbf{rcut1} \\ \mathbf{rcut1} &< r_{ij} < \mathbf{rcut2} \\ \mathbf{rcut2} &< r_{ij} < \mathbf{cutoff} \end{aligned} \quad (2.21)$$

where for Van der Waals interaction **cutoff** is substituted by the variable **rvdw** in the CONTROL file. The shells are handled via proper switching functions in

order to ensure continuity of forces at the shell borders. The switching functions are polynomials of order three governed by the “healing length” parameter **switch**.

The splitted propagator is of the form

$$\begin{aligned} & [\exp(i\frac{h_3}{2}L_3)[\exp(i\frac{h_2}{2}L_2)[\exp(i\frac{h_1}{2}L_1) \\ & \quad [\exp(ihL_o)]^{n_o} \\ & \quad \exp(i\frac{h_1}{2}L_1)]^{n_1} \exp(i\frac{h_2}{2}L_2)]^{n_2} \exp(i\frac{h_3}{2}L_3)] \end{aligned} \quad (2.22)$$

where h is the input time step and n_o , n_1 and n_2 are chosen in order to have the following time steps of increasing lengths

$$\begin{aligned} h_1 &= n_0 h \\ h_2 &= n_1 h_1 = n_1 n_0 h \\ h_3 &= n_2 h_2 = n_2 n_1 h_1 = n_2 n_1 n_0 h \end{aligned} \quad (2.23)$$

When using this splitting within the VV scheme and with the previously reported NVT and NPT dynamics, changes need to be applied with respect to the previous LF scheme. The reader is referred to the cited literature for a detailed explanation of technical points.

Switching between single and multiple time step molecular dynamics is achieved by changing the input parameters in the CONTROL file (see the **mult** directive in the CONTROL file). For example setting

$$\begin{aligned} & \mathbf{rcut1} \ 0.0 \\ & \mathbf{rcut2} \ 0.0 \\ & \mathbf{cutoff} \ 10.0 \\ & \mathbf{rvdw} \ 10.0 \\ & \mathbf{mult} \ 1 \ 1 \ 1 \end{aligned} \quad (2.24)$$

is a typical setting for a single time step simulation whilst

$$\begin{aligned} \mathbf{rcut1} & 7.0 \\ \mathbf{rcut2} & 9.0 \\ \mathbf{cutoff} & 10.0 \\ \mathbf{rvdw} & 10.0 \\ \mathbf{mult} & 2\ 3\ 2 \end{aligned} \tag{2.25}$$

is a possible choice for a multiple time step simulations. The multiple-time-step algorithm is handled by the routine **mloop**.

2.3 Scanning neighbors

DLPROTEIN retains the previous capabilities of DLPOLY to scan neighbors on a “all-pairs” (AP) basis by using a Brode-Alrich algorithm to order the interacting pairs and eventually split them on several processors.

A direct scanning method based on the Link Cell (LC) method [2] is also been preserved, but DLPOLY supported only cubic or orthorombic PBC for the LC method. The LC method allows a drastic raise in CPU efficiency for large systems since CPU time grows linearly with the system size (AP is quadratic).

Those scanning methods have been modified in order to

- implement AP and LC on scalar platforms and optimize CPU and memory use.
- extend LC to different PBC’s (truncated octahedron and dodecahedron).

Actually in the CONTROL file it is possible to chose whether to use AP or LC method. The LC method is applied with different methods if the run is scalar or parallel. In the case of a parallel run LC is activated only if it produces effective CPU optimization and this condition is true if

$$\mathbf{cutoff} \leq \min(Lx, Ly, Lz)/3 \quad (2.26)$$

With a scalar run the condition is less strict. If LC is not usable, eventually due to a cell shrinking as time proceeds with an isobaric run, automatic switching to AP scanning is performed and the user is informed.

2.3.1 Memory optimization on scalar workstation

DLPROTEIN has routines that optimize memory handling by substituting a 2d-array array for storing interacting pairs into a 1d-array. The method is suited

to run on scalar machines and a good speed-up is achieved with respect to other methods due to a better cache memory use.

An extra factor 2 in memory is gained by using the “INTEGER*2” definition for the interacting list. This is actually suited for number of atoms less than 32767, if this is not the case an error message is printed out and the code must be recompiled.

2.3.2 CPU optimization on scalar workstation

The actual version of DLPROTEIN handles the Link Cell method [2] for all possible boundary conditions implemented in the code. A different algorithm than in the original DLPOLY routine has been implemented. The routines that perform LC scanning on scalar workstation are **scalink_vlst_cube** and **scalink_respa_vlst_cube** for cubic and octahedral boxes, **scalink_vlst_ortho** and **scalink_respa_vlst_ortho** for general triclinic boxes.

2.4 The Smooth Particle Mesh Ewald method

DLPOLY used the Ewald method to compute the electrostatic interactions when using periodic boundary conditions [1]. The Ewald method splits the Coulombic interactions into two separate potentials in the form

$$U_c = \frac{1}{4\pi\epsilon_o} \sum_{n < j}^{N^*} \frac{q_j q_n}{r_{nj}} \operatorname{erfc}(\alpha r_{nj}) - \frac{1}{4\pi\epsilon_o} \sum_{\text{molecules}} \sum_{l \leq m}^{M^*} q_l q_m \left\{ \delta_{lm} \frac{\alpha}{\sqrt{\pi}} + \frac{\operatorname{erfc}(\alpha r_{lm})}{r_{lm}^{1-\delta_{lm}}} \right\} + U_{rec} \quad (2.27)$$

where U_{rec} is the reciprocal space term of the Ewald sum

$$U_{rec} = \frac{1}{2V_o\epsilon_o} \sum_{k \neq 0}^{\infty} \frac{\exp(-k^2/4\alpha^2)}{k^2} \left| \sum_j^N q_j \exp(-ik \cdot r_j) \right|^2 \quad (2.28)$$

as implemented in the **ewald1** routine of the original DLPOLY code and it is usually the main CPU-eater in biosimulations, moreover scaling as $N^{3/2}$ with the system size.

Alternatively DLPROTEIN can use the smooth particle mesh Ewald (SPME) method [7] to compute the reciprocal part of the Ewald sum. The SPME method does not compute U_{rec} as a sum over k-shells as for the original Ewald method, but it uses a three-dimensional grid to apply 3d-Fast Fourier Transforms (FFT) and obtain splined observables that reconstruct the requested energy term and corresponding forces. The SPME is called by the routine **ewald1pme**, a driver for the original SPME routines.

The SPME method requires as input parameters the order of the spline and the number of grid points in the x-, y- and z-directions. These parameters are specified in the CONTROL file as described in the following sections. Memory requirements must be satisfied by changing the MAXN, MAXORDER and MAXT parameters in the **ewald1pme** routine. If memory is necessary, i.e. for very dense grids or high order splines, the program will write the memory requirements and the code must be recompiled.

The SPME method is currently available only for general triclinic boxes so that it is forbidden for truncated octahedral and rhombic dodecahedral boxes.

SPME uses a public domain FFT routine or, alternatively, calls to libraries optimized on selected architectures (as Silicon Graphics and Cray machines, while the user can extend the calls to other platforms) The use of public domain or library calls to apply FFT is chosen in the compilation environment, as discussed in the section on the compilation directives. The parallel implementation of SPME is achieved by using the distributed routine “pccfft3d” to perform 3d-FFT, present on T3E and Silicon Graphics parallel architectures.

2.4.1 Choosing the SPME variables

The procedure for choosing the SPME control variables resembles the procedure usually applied in the standard Ewald sum method (e.g. see the DLPOLY manual). The only difference relies in substituting the number of $\mathbf{k}$ -vectors of the standard Ewald sum with the number of grid points of the SPME real space method. Furthermore in the SPME another factor can be varied so to achieve optimal convergence, this is the order of the splines used in the method. It is advised to use a well established and safe value of the spline order, say 6 or 8, and to focus on a good choice of the SPME variables, since the spline order does not affect the results if the spline order is large enough.

In the original Ewald method the “ewald precision” directive was available to find an optimal value of α and $\mathbf{k}_{max}$. The same directive can be used in order to find a good guess for α , but still a choice for the grid density must be taken. So, it is recommended to determine the α and $\mathbf{n}$ parameters manually, as described in the following (and as almost identically described in the DLPOLY manual).

Preselect the value for r_{cut} , choose a value for α of about $3.2/r_{cut}$ and a large value for $\mathbf{n}$ (say 50 50 50 or more). Then do a series of ten or so single

step simulations with your initial configuration and with α ranging over the value chosen plus and minus 20%. Plot the Coulombic energy (and the coulombic virial with opposite sign, $-W$) versus α . If the SPME is correctly converged you will see a plateau in the plot. Divergence from the plateau at small α is due to non-convergence in the real space sum. Divergence from the plateau at large α is due to non-convergence of the SPME term. Redo the series of calculations using smaller $\mathbf{n}$ values and by choosing the $\mathbf{n}$ parameters such that they can be only divided by 2, 3 or 5 (e.g. 50, 48, 45, ...). The optimum values for $\mathbf{n}$ are the smallest values that reproduce at the value of α to be used in the simulation the correct Coulombic energy (the plateau value) and virial. Keep in mind that the virial depends on forces and it converges more slowly than the energy - i.e. one usually requires a higher precision in the energy, say 10^{-6} , than in the virial.

Chapter 3

Other differences between DLPOLY and DLPROTEIN

3.1 Non-bonded interactions

In DLPROTEIN the actual number of available non-bonding potentials has been restricted in order to deal only with potentials usually employed in biological simulations (e.g. the GROMOS and CHARMM22 force fields). This is the Coulomb interaction (in the form bare $1/r$ form, shifted, reaction field or Ewald sum) and the Van der Waals interaction, both in the **12-6** and **lj** form for the input FIELD file.

3.2 Interpolation, spline or direct evaluation of the Van der Waals interactions

In order to increase performances DLPROTEIN can avoid the use of tabulated potential when computing the non-bonding forces. On the other hand splines can be used as a third alternative. Higher order interpolation schemes, such as the 4-points and r-square interpolation, as used in the original DLPOLY program, have

been eliminated. Interpolation (or spline) versus direct evaluation is managed in the CONTROL file. The directive “interpolate potentials” is used to indicate the use of an interpolation or a splining scheme. At the compilation level, when the compilation macro “SPLINE” is present the splined potential will be used instead of the 3-point interpolation. In the include file, specification of the grid points for the spline is determined by the parameter **mxspln**. The ratio between the grid points in a 3-point interpolation and spline potential can be up to 10 depending on the desired accuracy.

3.3 Refolding of molecules

An efficient way of reducing CPU time when dealing with molecules is to avoid molecule refolding in the primary cell of periodic boundary condition setting. Refolding means that molecules will be cut appear as cut in the primary cell. By avoiding refolding routines dealing with constraints will strongly ameliorate in performance. Refolding of molecules (as used originally in DLPOLY) is activated at the compilation level by the macro REFOLD, while by default the code will run without any refolding.

If refolding is absent the code will sew the molecules at step zero of the MD run in the routine **utaylor**. When a molecular species is constituted by blocks not connected by constraints or springs, refolding of molecules is mandatory.

3.4 Constraints treatment

In order to have exactly the same trajectories on scalar and parallel platforms, i.e. avoiding the different ways the shake algorithms is implemented in the `sca_shake` and `rdshake_1` routines, it is possible to force the code to use `sca_shake` also with parallel runs, where all processors will process all the constraints in the system.

This is achieved by specifying the macro SERIALSHAKE at the compilation level.

3.5 Shifted Van der Waals interactions and long range corrections

Better energy conservation is achieved when the Van der Waals potential is shifted at the chosen cut-off. The shift is activated through the **ljshift potential** directive in the CONTROL file.

The shift does not affect the forces but only the energy. Consequently the NVE trajectory of a shifted and unshifted potential will be exactly the same apart from a fluctuating term in the total energy. Consistently with the shifted potential model, the long range corrections to energy and pressure are set to zero. In this case the dynamics of the system is affected when performing NPT dynamics since in the unshifted system the long range correction to pressure is part of the pressure tensor and its act as a pseudo-force for the volume rearrangement.

3.6 Smoothing the Van der Waals interactions at the cut-off

A switching function can be used in order to eliminate discontinuities in the forces at the cut-off for the Van der Waals potential. Switching is obtained by a 3-rd order polinomial function that switches off the potential energy in the range $(rcut - dswch) < r < rcut$, where *dswch* is the healing length as specified in the CONTROL file by the directive **ljswitch**.

3.7 Hydrogen bond detection

In DLPROTEIN it is possible to analyse the presence of hydrogen bonds in the system during the run. The detection of hydrogen bonds is stored in the output file HBOND.dat in formatted (20i5) style, as the list of atomic indices of triplets of donor-hydrogen-acceptor.

Hbond formation is controlled by the directive **hbond** *n dist angle* in the CONTROL file, where *n* is the frequency for dumping the detected Hbonds, *dist* is the distance between donor and acceptor and *angle* is the angle formed between the vector connecting hydrogen-donor and the vector connecting acceptor to donor.

The specification of donors, polar hydrogens and acceptors in the system is specified in the FIELD file, as explained in the following.

3.8 Short and long OUTPUT files

When running biological molecules, i.e. when using very large FIELD files, the user may require that the OUTPUT file is shorter than the default value. A short or long version of the OUTPUT file can be obtained by using the directives **output short** and **output long** in the CONTROL file.

3.9 I/O format of trajectory data

A compact and formatted version of the dumped output trajectory is often useful with biomolecules. A formatted GROMOS-like version of the output data can be generated by using the directive **gromos dump** in the CONTROL file. In this case three different files, DUMPPOS.dat, DUMPVEL.dat, DUMPFOR.dat, are used to dump positions, velocities and forces separately. This is the default behaviour. When using the **history dump** directive the old style HISTORY file is recovered.

3.10 Include files

The main include file of DLPROTEIN is **dlprotein.inc** that should be modified for any change, in an analogous way to the **dl_params.inc** file for DLPOLY. In the utility directory the code **parset_dlprotein.f** has been added, analogous to the **parset** utility for DLPOLY.

A new include files has been added in DLPROTEIN that is used for declaring the arrays dimensions used in the SPME method. This file is called **darden.inc** (where its analogous for the T3E compilation is **darden_t3e.inc**).

3.11 T3E communication routine

A new version of the routines **gdsun** and **gisun** have been introduced to strongly reduce communication times on the Cray T3E parallel architecture.

3.12 Directives in the CONTROL file

In table 3.1 we report the directives in the CONTROL file that are changed or are not standard in the original DLPOLY2.0 package.

None of the added directives is mandatory. If not specified **rcut1**, **rcut2**, **rvdw** and the switching length are set to zero and the single time step scheme is applied.

directive:	meaning:
mult n_0 n_1 n_2	multiple time steps cycles. Given the timestep h , the RESPA timesteps are given by $h_1 = n_0 h$, $h_2 = n_1 h_1$, $h_3 = n_2 h_2$
rcut1 f	First cutoff (Å) for C+VdW interaction (region I of RESPA)
rcut2 f	Second cutoff (Å) for C+VdW interaction (region II of RESPA)
cutoff f	Last cutoff (Å) for C+VdW interactions (region III of RESPA or global interaction region with the single-time step scheme)
rvdw f	Maximum cutoff (Å) for VdW interactions. VdW are globally computed within the $(0, rvdw + swlen)$ sphere if $rvdw < cutoff$ and within the regions I, II, III, otherwise.
switch f	Healing length (Å) for switching off the non-bonding interactions at the region borders for RESPA.
output long	Verbose mode in OUTPUT file (default).
output short	Silent mode in OUTPUT file.
gromos dump	Three dumping files for trajectory, velocities and forces (Gromos style)
hbond n $dist$ ang	Hydrogen bond detection every n steps. Hbond is detected if the donor-acceptor distance is less than $dist$ (Å) and the hydrogen-donor-acceptor angle is less than ang degrees.
history dump	One history file for dumping the MD run (DLPOLY original style).
link cell	Activates the link cell method whenever the ratio between the cell d
ljshift	Shift the Van der Waals potential at the cut-off. Not shifted by default
ljswitch f	Smoothed Van der Waals potential at the cut-off in the region $rcut - f < r < rcut$. Unsmoothed by default.
interpolate potentials	Uses the r-interpolated or splined evaluation of the non-bonding potentials. Direct evaluation of VdW by default.
maxwell n	Global refresh of velocities from a Maxwellian every n steps (preferentially during equilibration).
spme α n n_x n_y n_z	select SPME ³⁵ method for electrostatics, with: α = Ewald convergence parameter (Å ⁻¹) n = spline order n_x = grid points in the x direction

<code>sitnam</code>	<code>a8</code>	atomic site name (unchanged)
<code>weight</code>	<code>f12.0</code>	atomic site mass (unchanged)
<code>chge</code>	<code>f12.0</code>	atomic site charge (unchanged)
<code>nrept</code>	<code>i5</code>	repeat counter (unchanged)
<code>ifrz</code>	<code>i5</code>	'frozen' atom (unchanged)
<code>igrp</code>	<code>i5</code>	neutral/charge group number (unchanged)
<code>hprop</code>	<code>a3</code>	Hbond propensity (added)

Table 3.2: Atomic sites specification in the FIELD file

3.13 Directives in the FIELD file

The only optional change in the FIELD file for DLPROTEIN is the specification of the donors, polar hydrogens and acceptors to detect Hbond formation in runtime. The format of the FIELD file regards the atomic species description, and appears as `(a8,2f12.0,3i5,a3)` (see table 3.13) The keyword `hprop` is equal to “`d.`” for a donor, “`.h.`” for a polar hydrogen attached to a donor, “`.a`” for an acceptor, “`d.a`” for an atom with both acceptor and donor propensity.

The mandatory rule to use when specifying polar hydrogens is that all polar hydrogens attached to a donor must follow in order the donor specification. Absence of a proper triplet will be set to “`...`” by default.

3.14 Directives in the Makefile

In this section we describe the Makefile directives added or changed with respect to the original DLPOLY2.0 package.

1. **VECTLIST**

This macro compiles the code to run with a 1-d array for storing the neighbors table. It is usable only with scalar architectures. It allows an optimal save in memory and CPU time. The arguments are:

- **VECTLIST** - 1-d array (default)
- **NOVECTLIST** - 2-d array

2. **STORE**

This macro compiles the code to run with a 2 bytes declaration of neighbor arrays. It allows a factor 2 in savings memory. Allowed only for number of atoms less than 32767. The arguments are:

- **STORE2** - INTEGER*2 declaration (default)
- **STORE4** - INTEGER*4 declaration

3. **FFT**

This macro uses a public domain or library versions of the Fast Fourier Transforms to apply the SPME method. The arguments are:

- **PUBFFT** - for serial public domain FFT (default)
- **SGIFFT** - for serial Silicon Graphics complib library
- **CRAYFFT** - for serial Cray libraries
- **PCCFFT** - for parallel library with the pccfft3d distributed routine available.

4. **SHAKE**

This macro forces the use of the standard shake algorithm to treat

constraints on parallel architectures (constraints are not distributed and all processors perform the same task). The arguments are :

- **SERIALSHAKE** - to force serial shake with parallel runs
- **NOSERIALSHAKE** - to use distributed shake algorithm (default).

5. **INTERPOLATE**

Switch of the calculation of potentials and forces between 3-point interpolation and cubic spline. Arguments :

- **SPLINE** - Spline potential
- **NOSPLINE** - 3-point interpolation (default).

6. **REFOLD**

Controls molecules refolding in the primary box when using periodic boundary conditions. Arguments :

- **REFOLD** - Molecules are refolded, i.e. broken, in the primary cell
- **NOREFOLD** - Molecules are not refolded (default).

3.15 Examples

A few examples are provided together with DLPROTEIN. The examples are kept in the “examples” directory and they are :

1. System composed of 500 argon atoms simulated for 1000 steps.

Ensemble : NVE

Van der Waals interaction : shifted + smoothed. Direct evaluation

No electrostatics

Scanning : all pairs basis

2. System composed of 256 SPC water molecules, 100 steps.

Ensemble : NVT (Nosé-Hoover)

Van der Waals interaction : shifted + smoothed. Splined potential

Electrostatics : SPME

Scanning : all pairs basis

3. System composed of 256 SPC water molecules, 200 steps.

Ensemble : NPT (Nosé-Hoover)

Van der Waals interaction : shifted + smoothed. Direct evaluation

Electrostatics : Ewald sum

Scanning : all pairs basis

4. System composed of Myoglobin + 1513 tip3p water molecules + 9 Cl counterions, 20 steps.

Ensemble : NPT (Nosé-Hoover)

Van der Waals interaction : shifted + smoothed. Direct evaluation

Electrostatics : SPME

Scanning : link cell method

In the “build-topology” directory an example of use of the build_all utility is provided. The example builds the topology of Myoglobin as used in the previous test case. The utility build_all can be run by answering a few questions or by using the prebuilt file with the command

```
build_all < input_build_all > out_build_all
```

A number of files are produced as execution proceeds. The output file FIELD.out is the global topology of Myoglobin + solvent + counterions. The topology of tip3p water and counterions are present as prebuilt FIELD files. The output files FIELD.1 and CONFIG.1 are the partial topology and configuration for Myoglobin. The CONFIG.1 file can be used in conjunction with the DLPOLY utility **water-add** to fill the MD box with water.

Bibliography

- [1] Smith, W., Forester, T.R. , “The DiPoly 2.0 User Manual” , T.R.Forester and W.Smith, CCLRC, Daresbury Laboratory, Daresbury, Warrington WA4 4AD, England (1995).
- [2] M.P.Allen e D.J.Tildesley, “Computer simulation of liquids” , Clarendon Press, Oxford, 1987.
- [3] Ciccotti, G., and Ryckaert, J.P., *Comp.Phys.Rep.***4**(1986), 345
- [4] Martyna, J.G., Tuckerman, M.E., Tobias, D.J., Klein, M.L., *Mol.Phys.***87**(1996), 1117
- [5] Melchionna, S., Ciccotti, G., Holian, B.L., *Mol.Phys.***78**(1993), 533
- [6] Procacci, P., Marchi, M., *J.Chem.Phys.***104**(1996), 3003
- [7] Essmann, U., Perera, L., Berkowitz, M.L., Darden, T., Lee, H., Pedersen, L.G. *J.Chem.Phys.***103**(1995), 8577